**BUCLES**

**B – I) – Bucles simples**

Son el tipo más básico su sintaxis es:

**LOOP**

*<Secuencia\_de\_órdenes>;*

**END LOOP;**

Este bucle seria infinito, no tiene condición de parada. Para salir de un bucle le pondremos la orden EXIT, que su sentencia es:

**EXIT [ WHEN** *<Condición>* **];**

Esta orden seria equivalente a:

IF <Condición> THEN

EXIT;

END IF;

Escribe un bloque pl/sql que permita calcular la suma de los 1000 primeros números naturales.

declare

i integer;

suma real;

begin

suma:=0;

i:=1;

loop

suma:=suma+i;

i:=i+1;

exit when i>1000;

end loop;

dbms\_output.put\_line('la suma de los 1000 primeros numeros naturales es '||suma);

end;

RESULTADO:

la suma de los 1000 primeros numeros naturales es 500500

**B – II) – Bucles WHILE**

Su sintaxis:

**WHILE** *<Condición>* **LOOP**

*<Secuencia\_de\_órdenes>;*

**END LOOP;**

De esta forma, siempre antes de entrar en el bucle evalúa la condición, si es verdadera entrará. Si la condición es falsa o nula el bucle se termina. Por eso hay que tener en cuenta que si la condición del bucle no toma el valor TRUE la primera vez que se le comprueba el bucle no llegará nunca a ejecutarse.

Pueden usarse las órdenes EXIT o EXIT WHEN dentro de un bucle WHILE para salir del bucle, sin llegar a terminar le Condición.

Escribe un bloque pl/sql que permita calcular la suma de los 1000 primeros números naturales.

declare

i integer;

suma real;

begin

suma:=0;

i:=1;

while i<=1000 loop

suma:=suma+i;

i:=i+1;

end loop;

dbms\_output.put\_line('La suma de los 1000 primeros numeros naturales es '||suma);

end;

RESULTADO:

La suma de los 1000 primeros numeros naturales es 500500

**B – III) – Bucles FOR**

En el caso en que sepamos el número de iteraciones en que se ejecutarán los bucles simples y while utilizaremos los bucles FOR, su sintaxis es:

**FOR** *<Contador\_bucle>* **IN [ REVERSE ]** *menor … mayor* **LOOP**

*Secuencia\_de\_órdenes*

**END LOOP;**

Donde <Contador\_bucle> es una variable que no hace falta que se declare ya que lo hace de forma implícita. Los valores menor…mayor muestra el rango en que se ejecutará el bucle.

Escribe un bloque pl/sql que permita calcular la suma de los 1000 primeros números naturales.

declare

i integer;

suma real;

begin

suma:=0;

for i in 1 ..1000 loop

suma:=suma+i;

end loop;

dbms\_output.put\_line('La suma de los 1000 primeros numeros naturales es '||suma);

end;

RESULTADO:

La suma de los 1000 primeros numeros naturales es 500500

**C ) – GOTO Y ETIQUETAS**

**GOTO**  *Etiqueta*;

declare

i integer;

suma real;

begin

suma:=0;

dbms\_output.put\_line('LLEGA');

goto resultado;

dbms\_output.put\_line('NO LLEGA');

for i in 1 ..100 loop

suma:=suma+i;

dbms\_output.put\_line('NO ENTRA');

end loop;

<<resultado>>

dbms\_output.put\_line('La suma de los 1000 primeros numeros naturales es '||suma);

end;

RESULTADO:

LLEGA

La suma de los 1000 primeros numeros naturales es 0

**C – I) – Restricciones de GOTO**

-No puede haber otra etiqueta en el entorno actual con el mismo nombre.

-La etiqueta debe preceder a un bloque o a un conjunto de órdenes ejecutables.

-No se puede saltar al interior de un bucle

-No se puede saltar al interior de una orden IF

##### C – II) – Etiquetado

A los bucles pueden ponérseles etiquetas de forma que las usemos en la sentencia EXIT. En el caso de que se le añada una etiqueta a un bucle habrá que ponerla también al final del bucle.

Escribe un bloque pl/sql que permita calcular la suma de los 1000 primeros números naturales.

declare

i integer;

suma real;

begin

suma:=0;

i:=1;

<<mi\_bucle>>

while i<=1000 loop

suma:=suma+i;

i:=i+1;

end loop mi\_bucle;

dbms\_output.put\_line('la suma de los 1000 primeros numeros naturales es '||suma);

end;

RESULTADO:

la suma de los 1000 primeros numeros naturales es 500500

**D) – ORDEN NULL**

En algunos casos, podríamos querer indicar explícitamente que no se realice ninguna acción. Esto puede hacerse mediante la orden NULL, que es una orden que no tiene ningún efecto.

Realizar un bloque pl/sql que calcule el valor absoluto de un número negativo.

declare

i integer:=&i;

begin

if i<0

then

dbms\_output.put\_line('el valor absoluto de '||i||' es: '||-i);

else

null;

end if;

end;

**EJEMPLOS DE BUCLES**

**--BUCLE PARA MOSTRAR LOS NUMEROS ENTRE 1 Y 10**

**--1) BUCLE WHILE**

**declare**

**i *int*;**

**begin**

**i := 1;**

**while i <= 10 loop**

**dbms\_output.put\_line(i);**

**i := i + 1;**

**end loop;**

**dbms\_output.put\_line('Fin de bucle while');**

**end;**

**--2) BUCLE FOR**

**declare**

**begin**

**for i in 1..10 loop**

**dbms\_output.put\_line(i);**

**end loop;**

**dbms\_output.put\_line('Fin de bucle For');**

**end;**

**--PEDIR AL USUARIO UN NUMERO INICIO &inicio**

**--Y UN NUMERO FINAL**

**--MOSTRAR LOS NUMEROS COMPRENDIDOS ENTRE DICHO RANGO**

**--SI EL NUMERO INICIAL ES MAYOR, LO INDICAMOS Y NO HACEMOS EL BUCLE.**

**declare**

**inicio *int*;**

**fin *int*;**

**begin**

**inicio := &inicial;**

**fin := &final;**

**--preguntamos por los valores de los numeros**

**if (inicio >= fin) then**

**dbms\_output.put\_line('El número de inicio (' || inicio ||**

**') debe ser menor al número de fin (' || fin || ')');**

**else**

**for i in inicio..fin loop**

**dbms\_output.put\_line(i);**

**end loop;**

**end if;**

**dbms\_output.put\_line('fin de programa');**

**end;**

**undefine inicial;**

**undefine final;**

**--QUEREMOS UN BUCLE PIDIENDO UN INICIO Y UN FIN**

**--MOSTRAR LOS NUMEROS PARES COMPRENDIDOS ENTRE DICHO INICIO Y FIN**

**declare**

**ini *int*;**

**fin *int*;**

**begin**

**ini := &inicial;**

**fin := &final;**

**for i in ini..fin loop**

**if (mod(i, 2) = 0) then**

**dbms\_output.put\_line(i);**

**end if;**

**end loop;**

**dbms\_output.put\_line('Fin de programa');**

**end;**

**undefine inicial;**

**undefine final;**

**--CONJETURA DE COLLATZ**

**--La teoría indica que cualquier número siempre llegará a ser 1**

**--siguiendo una serie de instrucciones:**

**--Si el número es Par, se divide entre 2**

**--Si el número es Impar, se multiplica por 3 y sumamos 1**

**--6,3,10,5,16,8,4,2,1**

**declare**

**numero *int*;**

**begin**

**numero := &valor;**

**while numero <> 1 loop**

**--AVERIGUAMOS SI ES PAR/IMPAR**

**if (mod(numero, 2) = 0) then**

**numero := numero / 2;**

**else**

**numero := numero \* 3 + 1;**

**end if;**

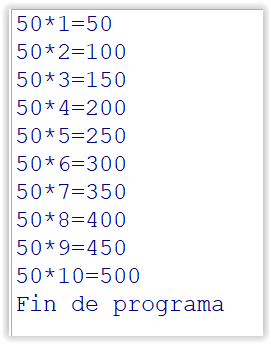
**dbms\_output.put\_line(numero);**

**end loop;**

**dbms\_output.put\_line('Fin de programa');**

**end;**

Mostrar la tabla de multiplicar de un número que pidamos al usuario.



**declare**

**numero *int*;**

**operacion *int*;**

**begin**

**numero := &valor;**

**for i in 1..10 loop**

**operacion := numero \* i;**

**dbms\_output.put\_line(numero || '\*' || i || '=' || operacion);**

**end loop;**

**dbms\_output.put\_line('Fin de programa');**

**end;**

Quiero un programa que nos pedirá un texto.

Debemos recorrer dicho texto letra a letra, es decir, mostramos cada letra del texto de forma individual.

**declare**

**v\_texto *varchar2*(50);**

**v\_longitud *int*;**

**v\_letra *varchar2*(1);**

**begin**

**v\_texto := '&texto';**

**--UN ELEMENTO EN ORACLE EMPIEZA EN 1**

**--en un lugar de la mancha...**

**v\_longitud := length(v\_texto);**

**for i in 1..v\_longitud loop**

**v\_letra := substr(v\_texto, i, 1);**

**dbms\_output.put\_line(v\_letra);**

**end loop;**

**dbms\_output.put\_line('Fin de programa');**

**end;**

Necesito un programa dónde el usuario introducirá un texto numérico: **1234**

Necesito mostrar la suma de todos los caracteres numéricos en un mensaje.

La suma de **1234** es **10**

**declare**

**v\_texto\_numero *varchar2*(50);**

**v\_longitud *int*;**

**v\_letra *char*(1);**

**v\_numero *int*;**

**v\_suma *int*;**

**begin**

**v\_suma := 0;**

**v\_texto\_numero := &texto;**

**v\_longitud := length(v\_texto\_numero);**

**for i in 1..v\_longitud loop**

**v\_letra := substr(v\_texto\_numero, i, 1);**

**v\_numero := to\_number(v\_letra);**

**v\_suma := v\_suma + v\_numero;**

**end loop;**

**dbms\_output.put\_line('La suma de ' || v\_texto\_numero || ' es ' || v\_suma);**

**end;**